自然言語処理 58 － 3
（１９８６．１１．２１）

グラフのマッチングを用いた意味解析

丸山 宏

日本アイ・ビー・エム株式会社 東京基礎研究所

自然言語理解においては、構文情報が得られない場合でも意味情報だけから何かしらの意味を組み立てられる意味解析のメカニズムをもつことが重要である。我々はグラフのマッチングに基づく意味解析の手法を提案する。この方法を用いれば構文情報が与えられない場合でも与えられた意味モデルの範囲において、必ず意味のある意味構造が得られる。また、適当なスコアリング関数を定義することによって、いくつかの意味のある解釈の中からもっともらしいものを選ぶことができる。さらに、構文的な情報が得られる場合には、それを制約またはヒントとして用いることによって、より候補を絞ることができることを示す。

Semantic Analysis Using Graph Matching

Hiroshi MARUYAMA

Tokyo Research Laboratory, IBM Japan Ltd.
5-19, Sanbancho, Chiyoda-ku, Tokyo 102, Japan

A semantic analyzer of a natural language understanding system should be capable of constructing some semantic interpretation even if no syntactic information is provided. We propose a semantic analyzing mechanism with this capability using a graph-matching technique. It is shown that if the meaning of every input word is valid in a given semantic model, the result of semantic analysis is also valid with respect to that model. Furthermore, our technique provides a mean for specifying the plausibility of each possible interpretation. Syntactic information can be used as hints for this disambiguation process.

１．はじめに

自然言語を理解する上で、しばしば、構文的な情報が意味の抽出にほとんど役に立たない場面がある。日本語に多い、『～の～』という形はその典型的な例である。たとえば、『ＪＡＬのフライト』といったら『航空会社がＪＡＬであるフライト』の意味であるし、『今日のフライト』といったら『出発日時が今日であるフライト』の意味となる。どちらの場合も『ＪＡＬ』と『フライト』あるいは『今日』と『フライト』の間の関係をつけるためには、構文上の助詞『の』はほとんど役に立っていないといえる。極端にいえば、『ＪＡＬ、フライト、ありますか』のような文でも、立派に意味を伝え得るのである。複合語の解析も同じ問題を含んでいる。たとえば、『情報処理機械』と『情報処理学会』の係り受けのちがいは意味情報をフルに使わなければできない。

また、自然言語によるＱＡシステムのように、ユーザーからの構文情報が完全には信頼できない場合もある。会話においては、より多くの省略や、文法的な不適格文すら使われるからである。

一方、単語間の意味関係が自明ならば、構文情報はそもそも必要ないという考え方もできる。もしドメインが限られたＱＡシステムなら、『フライト、ＪＡＬ、ありますか？』でも、『ＪＡＬ、ありますか？、フライト』でも同様に解釈できることが要求されるであろう。

このように考えてくると、自然言語理解システムにとっては、構文情報が得られない場合においても、何かしらの意味が組み立てられる意味解析の基本的なメカニズムが必要であることがわかる。このメカニズムは、与えられたいくつかの単語（あるいは句）の意味表現から、それらを有機的につなげて一つの意味のあるまとまりを作るのである。本論文ではこのような意味解析のメカニズムの一つを提案する。このメカニズムはＸＣＧと呼ばれる、ラベル付きのグラフのマッチング操作に基づいていて、意味的に可能な解釈だけを残らず生成する。生成された意味構造にはもっともらしさを示すスコアがついているので、その中から文脈やその他の条件を加味した上で、正しいと思われる解釈を選ぶことができる。

もちろん、構文的な情報を使わないと、少し複雑な文では正しい解釈が得られない。『太郎、次郎、お金、あげた』では、『太郎が次郎にお金をあげた』なのか、その逆なのかがあいまいである。我々は構文情報を、先に述べた意味解析メカニズムへのヒントとして捉えることにする。グラフのマッチング・メカニズムに対しては、いくつかのマッチングに関する制約条件を、構文情報からのヒントとして入力する。

２節では、既存の自然言語処理の手法について、基本的な意味解析メカニズムと、それに対して構文情報がヒントとしてどのように使われているかという２点から考察する。３節ではＸＣＧとその上で定義されるマッチングを用いた意味解析について述べる。構文情報の使い方については４節で、また、いくつかの残された問題点については５節で議論する。

２．今までの手法

今までに多く提案されてきた、意味解析の手法の主なものについて、どのように個々の単語の意味を組み合わせるかという点と、その際に構文情報がどのように使われるかを考えてみよう。構文情報としては、どの単語（または句）の意味を先にくみあわせるかという順序に関する情報（これを構文情報Ｉと呼ぶことにする）と、その組み合わせ方に対するヒント、すなわちどちらが主語でどちらが述語かという情報など（これを構文情報IIと呼ぶことにする）にわけられる。句や節といった、構文要素はひとかたまりの意味を表わすという考え方に基づけば、構文情報Ｉ（順序に関する情報）は解析の手法によらず、意味解析の順序のコントロールという、同じ使われ方をする。したがって、ここでは、構文情報IIの使われ方に注目する。

自然言語の意味表現として最も多く用いられているものはフレーム[Minsky75]の考え方をベースにしたものであろう。この考え方では、あるひとかたまりの意味はフレームすなわち＜スロット名、値＞のペアのセットで表わされる。値としては、通常、アトミックな意味を表わすシンボルのほかに他のフレームもとることができて、したがってフレームによる意味表現はネットワーク状の大きなデータ構造をなすことができる。フレームに基づく意味構成の基本的な操作はスロットへの値の代入(destructive assignment)である。例えば『太郎は走る』の『太郎』と『走る』のそれぞれの意味を合成するには、『走る』を表わすフレームの主格スロットに『太郎』を表わすフレームを値として代入する。スロットには通常、デフォールト値やとりうる値の範囲などを指定したり、値が代入あるいは参照された時に起動される手続きを付加したりすることができて、このことによって意味的におかしいものを排除したり、明示的に言われていないものを推察したりすることができる。

近年は論理プログラミングの普及にともない、フレームの値の束縛のしかたを代入ではなく、単一化(unification)で行なうことが注目されている[田中86]。このやりかたは、ＬＦＧ、ＧＰＳＧなどの文法理論と相性が良いことが示されており[三吉85][三吉86]、また、異なるフレーム間の値の共有や不完全な値どうしのマッチングが可能などの、優れた特徴をもっている。

しかし、代入もしくは単一化のいずれの操作を行なう場合にも、どれが値を束縛されるべきスロットで、どれが値となるべきものかということがわかっていないと、意味構造を構築することができない。フレームをベースとする意味表現を用いる場合には、まさに、このために構文情報IIが使われるのである。

ヤチマタ[藤崎79]、Chat-80[Pereira83]などデータベース照会のための自然言語インターフェースには、述語論理をベースにする意味表現がよく用いられる。一階述語論理は関係データベースと相性が良いためであろう。述語論理による意味表現における意味構成の基本操作は、論理変数どうしの単一化と、&、|、¬などのオペレータを用いた論理式の組み合わせである。たとえば『太郎は食べた』においては『太郎』(human(X)&X=太郎)と『食べた』(eat(X',Y,past))のXとX'を単一化し、それぞれの論理式を&でつなげると必要な意味構造が得られる。この場合、構文情報IIはどの変数とどの変数を単一化するか、およびどのオペレータを選ぶかを指定する。

Montague文法[Montague74]では単語の意味はλ式で記述される。意味の組み合わせ方はλ式の適用(application)である。つまり、二つのλ式の片方を引数として他方を関数として適用するのである。この場合構文情報IIはどちらが関数で、どちらが引数であるかを指定するのに用いられる。例えば、主語と動詞では、主語が関数となり、動詞が引数となる。

その他にも意味解析の手法は数多く報告されているが、それらの多くはアドホックで、手続き的なものに近い。例えば、SchankのMERGIEシステム[Schank81]にみられる意味解析は、各単語毎に手続きを付加して、その手続きによって各単語独自の意味解析を行なっていく。

以上のようなそれぞれの意味表現について、意味構造の構築のための基本的操作と、構文情報IIの使われ方を表1に示す。いずれの場合も、構文情報IIが得られなかった場合に意味をどのように構築していくかという一般的な原則は与えられていない。

意味表現	意味構成の基本的操作	構文情報IIの使われ方
フレーム	代入(destructive assignment) 単一化(unification)	スロット及び値の選択
述語論理	変数の単一化 論理オペレータ	変数の選択及びオペレータの選択
Montague文法	λ式の適用	関数と引数の区別

表 1. 意味構成の手法の比較

3. XCGによる意味解析

我々は既にConceptual Graph[Sowa84]を拡張した自然言語の意味表現（これをXCGと呼ぶ）を考案し、これを用いて談話における文と文脈との関係を解析する手法を考案した[Maruyama86][丸山86a][丸山86b]。このアルゴリズムは文と文脈を表わす二つの意味構造を入力とし、それらを意味的なつながりを考慮して組み合わせ、より大きな新しい文脈を表わす構造を生成する。文脈解析においては、文と文とのつながりを示す構文的な手がかりが少ないために、単語と単語（あるいは句と句）の場合における構文情報IIに相当する情報が得られにくい。このため、我々の文脈解析のアルゴリズムは、文と文との構文的な関係はほとんど使わない。したがって、逆にこの同じメカニズムが文中の意味の解析にも使える可能性がある。本節では、XCG並びにその意味解析への適用について述べる。

XCGはラベル付きの意味ネットワークの一種である。XCGは概念と呼ばれるノードと、関係と呼ばれる有向アークからなる。一つの概念はタイプT、名前r、注目度aの3つ組であり、[T:r,a]と表記される。たとえば、『浩一は飲む』をXCGで表わすと次のようになる。

[BOY:浩一,2]←(AGNT)←[DRINK:*,1.5]

タイプはその概念の属するクラスである。タイプはラティスになっていて、複数のスーパータイプをもつことを許す。名前はその概念につけられたユニークなシンボルである。*は名前の定まっていない概念につけられる一時的な名前

である。注目度はその概念がどの程度注目されているかを表わし、概念どうしのマッチングがおきた場合に、そのマッチングがどの程度もっともらしいかを計算するのに用いられる。以下、特に必要のない場合には、記法として、注目度と*については省略してもかまわないものとする。

自然言語の入力をある自然言語インターフェースが理解するということは、その自然言語の入力を応用システムが理解できる形に変換するということである。したがって、自然言語インターフェースシステムにおいては、まず応用システムが理解できる意味の範囲を規定しなければならない。これを意味モデルと呼ぶことにする。ＸＣＧにおいては、意味モデルMは

1）タイプのセット：Ω、ただし　$\perp \in \Omega$、$\top \in \Omega$（$\perp$はすべてのタイプのサブタイプ、$\top$はすべてのタイプのスーパータイプ
2）名前のセット：Ｉ、ただし　$* \in I$　（*は不定の名前）
3）それぞれの名前にそのタイプを対応させる関数type：$I \rightarrow \Omega$、ただし $type(*)=\perp$
4）各タイプ$T \in \Omega$についてのタイプ定義

で与えられる。タイプＴの定義は次の二つの項目からなる。

1）そのタイプのスーパータイプ：super(T)
2）そのタイプの概念に接続されうる外向きアークの定義：$arc(T)=\{\langle R_1,T_1^*\rangle, \langle R_2,T_2^*\rangle, \cdots, \langle R_n,T_n^*\rangle\}$

タイプT_1がタイプT_2のサブタイプであるとき、$T_1 \leqq T_2$とあらわす。一般にタイプの集合は$\leqq$に関して半順序をなす。外向きアークの定義は$\langle R,T^*\rangle$というペアのセットであらわされる（但し、$R_i \neq R_j$ if $i \neq j$）。タイプＴの外向きアークの定義の中に$\langle R_1,T_1\rangle$というペアがあるということは、タイプＴの概念に対して、

[T]→(R)→[T']（ただし、$T' \leqq T^*$）

というアークが高々一個存在してもよいことを示す。外向きアークの定義はそのスーパータイプの定義から継承される。したがって、あるタイプT_1の外向きアーク定義$arc(T_1)$の中に$\langle R,T_1^*\rangle \in arc(T_1)$なる要素があるならば任意のサブタイプ$T_2$（$T_2 \leqq T_1$）の外向きアーク定義$arc(T_2)$の中に$\langle R,T_2^*\rangle$（但し$T_2^* \leqq T_1^*$）なる要素が含まれていなければならない。

例として、タイプACTとタイプSAYの定義を示す。

```
super(ACT)=T
arc(ACT)={<AGNT,ANIMATE>}
super(SAY)=ACT
arc(SAY)={<AGNT,HUMAN>,<OBJ,INFORMATION>}
```

ただし、HUMAN≦ANIMITEと仮定する。

ある意味モデルMが与えられた時、あるグラフＧがその意味モデルに照らして意味をなすかという判断は、次のcanonicalの定義で与えられる。

定義(canonical graph)：グラフＧはその意味モデルＭに関して次の条件をみたすとき、canonicalであるという。
1）$\forall c_1:[T_1:r_1,a_1]$, $c_2:[T_2:r_2,a_2]$に対して$r_1=r_2 \neq *$ならば$c_1=c_2$
2）$\forall c:[T:r,a]$に対してtype(r)≦T
3）$\forall$アーク$A_1:c_1 \rightarrow (R) \rightarrow c_2$ $(c_1:[T_1:r_1,a_1], c_2:[T_2:r_2,a_2])$に対して $\exists T_2^*:\langle R,T_2^*\rangle \in arc(T_1)$かつ$T_2 \leqq T_2^*$。また、$A_2:c_1 \rightarrow (R) \rightarrow c_3$ $(c_2 \neq c_3)$となるA_2が存在しない。

意味モデルは応用システムが入力として理解できる意味表現を規定するという考えに立てば、すべてのcanonicalなグラフは、応用システムにとって意味をもつ。後に述べるグラフのマッチングの操作は、入力グラフがcanonicalである限り、必ずcanonicalなグラフを生成することが保証される。このことが、構文情報の助けを借りずに意味を生成する上で重要な点となる。

グラフのマッチング操作を定義する前にグラフの特化(specialization)を定義する。グラフｕがグラフｖの特化である（ｕ≦ｖと記す）ということは、次のうちのいずれかをみたすということである。

定義(specialization)：

1）u=v

2）u'≦v、但し、u'はvのある概念[T:r,a]を[T':r,a](T'≦T)でおきかえたもの。但し、type(r)≦T'でなければならない。

3）u'≦v、但し、u'はvのある概念c_1:[T:*,a]を[T:r,a]でおきかえたもの。但し、type(r)≦T。もしこのときに、uの中に同じ名前を持つ別な概念c_2:[T':r,a']がある場合にはu'に対して次に述べるfolding操作をc_1,c_2の上で行なわなくてはならない。

定義(folding)：あるグラフuの概念c_1,c_2をfoldingするとは、次のことをいう。

1）c_1とc_2の最大共通特化（GCS）を求めc_0とする。GCSがみつからない場合、foldingは失敗する。

2）c_1とc_2を終点にもつすべてのアークの終点をc_0に書き換える。

3）c_1とc_2からでるすべてのアークの始点をc_0に書き換える。

4）もし、c_0から同じラベルをもつリンクr_1，r_2がでている場合はr_1，r_2の終点どうしをfoldingし、リンクを１本にする。このfoldingができない場合は全体のfoldingが失敗する。

定義(GCS)：概念c_0:[T_0:r_0,a_0]は次の二つの条件を満たすとき、概念c_1:[T_1:r_1,a_1]と概念c_2:[T_2:r_2,a_2]の最大共通特化(GCS)という。

1）T_0=min(T_1,T_2)

2）r_1=r_2=r_0 または r_1=*, r_2=r_0 または r_2=*, r_1=r_0

特化の例：　たとえば、『浩一はおたべを食べる』

[BOY:浩一]←(AGNT)←[EAT]→(OBJ)→[おたべ]

は『人が何かをたべる』

[HUMAN]←(AGNT)←[EAT]→(OBJ)→[FOOD]

の特化である。

特化の定義のしかたから、次の定理が導かれる。

定理１：あるグラフuがcanonicalならば、そのグラフの特化u'(u'≦u)もcanonicalである。

グラフuの特化u'とグラフvの特化v'が共通の(連結)サブグラフwをもち、しかもu'-wとv'-wに共通な名前をもつ概念が存在しないとき、グラフu'とグラフv'をwの上で重ねて、一つの大きなグラフGを作ることが可能である。このマッチングの操作をjoinとよぶ。

またこうしてできたグラフGのことをuとvのjoinと呼び、join(u,v)とあらわす。定理１とjoinの定義から次の定理２が導かれる。

定理２：グラフu,vがcanonicalならば、join(u,v)もcanonicalである

この定理は、joinの操作を用いてある意味モデルM上で意味のあるグラフどうしを組み合わせた場合、得られたグラフはモデルM上で意味をなすことを保証している。

例として、『浩一は車を買った』というグラフu

[BOY:浩一]←(AGNT)←[BUY]→(OBJ)→[CAR]

と『彼の買った車は赤い』というグラフv

[HUMAN]←(AGNT)←[BUY]→(OBJ)→[CAR]→(COLOR)→[RED]

とのjoinを考えよう。uの特化u'としてuそのもの、vの特化v'として

[BOY:浩一]←(AGNT)←[BUY]→(OBJ)→[CAR]→(COLOR)→[RED]

をとると、共通のサブグラフw

[BOY:浩一]←(AGNT)←[BUY]→(OBJ)→[CAR]

ができる。このサブグラフの上で、u'とv'を重ねると、join(u,v)は

[BOY:浩一]←(AGNT)←[BUY]→(OBJ)→[CAR]→(COLOR)→[RED]

となる。

一般にグラフu,vのjoinの作り方は、それらの特化u',v'の作り方と共通なサブグラフwの選び方によって変わってくる。与えられたｕ，ｖに対して考えられる複数のjoinの間の関係について考察してみよう。以下、グラフｕ，ｖの二つのjoinを、$join_1(u,v)$、$join_2(u,v)$、また、$join_1$を作る際のu,vの特化および選ばれた共通サブグラフをそれぞれu_1',v_1',w_1、$join_2$を作る際の対応するグラフをそれぞれu_2',v_2',w_2とする。joinの間の関係$\leqq_J$を次のように定義する。

定義($\leqq_J$): $join_1 \leqq_J join_2$ <==> $w_1 \subset w_2$かつ$u_1' \leqq u_2'$かつ$v_1' \leqq v_2'$

この関係は、共通なサブグラフが大きければ大きいほど、また、joinを作る際に行なう特化が少なければ少ないほど、良いjoinであることを述べている。定義から明らかに関係$\leqq_J$は半順序をなすので、あるｕ，ｖが与えられた時に、順序$\leqq_J$に関してmaximalなjoinというものを考えることができる。maximalなjoinは、maximalでないjoinに比べて、より少ない仮定（特化）でより強い関係（共通グラフ）を得たものだということができる。

maximalなjoinは、しかし、一般にはやはり複数個存在することが考えられる。このために、ｕ，ｖの中にあらわれる注目度を利用して、それぞれのmaximal joinに対してヒューリスティックなスコアを与えることができるようになっている。このスコアの与え方の一例は文献［丸山86b］に述べられている。

構文情報が得られない時にjoinがうまく使える例として『浩一のオフィス』という場合を考えよう。しかし、与えられた意味モデルの中では、オフィスは部に付く属性であって、従業員に付く属性ではなかったとする。そうすると、これは本当は『浩一の部のオフィス』というべきところである。だが、『浩一』と『オフィス』を結ぶ助詞『の』だけではそれだけの情報は得られない。さて、『浩一』はある会社の従業員であるので、『浩一』をあらわす意味表現には、氏名や生年月日などの他にも会社での所属をあらわす情報が付いている。例えば、

[EMP:浩一]→(生年月日)→[DATE:1958/12/8]

.

.

→(所属)→[部:営業]

となる。一方、『オフィス』は『部』についている属性なので、例えば

[ROOM]←(OFFICE)←[部]

と表わされるだろう。ここで上に述べたjoinを行なえば

[EMP:浩一]→(所属)→[部:営業]→(OFFICE)→[ROOM]

という関係が把握できるのである。

４．構文情報IIの使い方

3節では構文情報が得られない場合、2つの意味表現から、もっともらしい意味を組み立てる方法について述べた。本節では、2つの意味表現が例えばそれぞれ主語と述語を表わすことがわかっている場合に、その情報を、複数のmaximal joinの候補の中から正しいものを選ぶのに、どのように用いるかについて述べる。

まず、グラフの主概念(head)という考えを導入する。主概念はある単語あるいは句が複数の概念からなるグラフであらわされた時に、意味の中心をなす概念である。たとえば『赤い車』の主概念は『車』となる。さて、もし構文的な情報からある句Aの中心的意味が別の句Bを修飾するという関係が明らかだった場合には（たとえば『浩一が食べる』の『浩一が』が『食べる』を修飾することが明らかであるような場合には）、句Aと句Bのjoinにおいて、句Aの主概念が必ず句Bのいずれかの概念と重ね合わさる筈であるといってよいだろう。このヒントをjoinの際に考慮することによって、かなりの数のありそうもないjoinの可能性を排除することが可能になる。このタイプのjoinを、→x型のjoinとよぶ。できあがった句の主概念は句Bの主概念となる。

『彼は部長だ』のように2つの句が同一のものを指していることが構文上明らかな場合は、もっと強い制約、すなわち、句Aの主概念と句Bの主概念が重なるという制約をおくことも可能となる。このタイプのjoinをxx型のjoinと呼ぶ。

先ほどの『浩一のオフィス』のように係り受け関係が構文上からははっきりしない場合には、このような制約をおくことはできない。したがってヒントなしでjoinを行なわなくてはならない。ヒントのないjoinを--型のjoinと呼ぶ。この際の全体の主概念は被修飾句の主概念、したがってこの場合は『オフィス』の主概念となる。

『浩一は京都から来た』というときのように、助詞の『から』はそれが起点格であることを常に示す。ＸＣＧでは意味を構築する手段としてjoinしか与えられていないので、この『から』の意味もグラフで表現する必要がある。このような助詞の意味は

[ENTITY:*y]←(SRCE)←[ACT:*x]

のように2つの主概念をもつグラフとして表わす。主概念*yは修飾句の主概念と重ね合せられ、全体としての主概念は*xとなる。このタイプのjoinをxyy型のjoinと呼ぶことにする。xyy型のjoinは、必ず第一の引数として主概念を2つ持つグラフをとる。このようにすると、『京都から』に対する意味表現は

[LOCATION:京都]←(SRCE)←[ACT:*x]

となる。

一方、格助詞『が』は、『浩一が食べる』というときのように、修飾句が動作主であることを強く示唆しているが、もちろん、これはすべての場合にあてはまるのではない（『浩一はおたべが好きだ』の『が』を考えてみよ）。このことが、構文に強く依存する既存の多くの自然言語理解システムを複雑なものにしている。ＸＣＧでは注目度をうまく使うことによって、このような選好を表現することが可能である。すなわち、『が』の意味を

[ENTITY:*y]←(AGNT)←[ACT:*x,3]

[ENTITY:*y]←(OBJ)←[ACT:*x,2]

という2とおりのグラフで表わす。このようにしておけば、『が』の意味として第一の意味が選ばれた場合には、（joinのスコアリング関数の定義にもよるが）第二の意味が選ばれたときよりもより高いスコアを得ることができる。

５．おわりに

自然言語理解においては、構文情報が得られない場合でも意味情報だけから何かしらの意味を組み立てられる意味解析のメカニズムの重要性を論じ、グラフのマッチングから正しい意味構造を構築する方法を提案した。この方法を用いれば与えられた意味モデルの範囲において、必ず意味のある意味構造が得られる。また、適当なスコアリング関数を定義することによって、いくつかの候補の中からもっともらしい解釈を選ぶことができる。さらに、構文的な情報が得られる場合には、それを制約またはヒントとして用いて、より候補を絞ることができることを示した。

ＸＣＧはある概念をそれに連想される他の概念とのネットワーク構造で表現するという点で、フレームによく似ているが、スロットへの値の代入ではなくてグラフどうしのマッチングが定義されているので、フレームよりもよりグローバルな意味の制約を表現することができ、また、このようにして得られた意味構造が与えられた意味モデルの中

で必ず意味をなすことが保証されているのが大きな利点である。単一化の概念は２つの概念間の最大共通特化によって、より制限されたかたちで導入されている。

ＸＣＧは以上のような利点をもつが、現在の定義のままでは、表現できない（あるいは表現することが困難な）ものもいくつかある。ひとつは、限量（quantification)の問題である。ＸＣＧにおける不定の名前をもつ概念は論理学でいえば存在の限量を表わす。ところが、これに全称や否定の限量を導入しようとすると、それらのスコープの問題が生じてくる。１つのグラフの『あるサブグラフに対して』といういいかたがＸＣＧでは与えられていないためである。[Sowa84]ではConceptual GraphにPROPOSITIONという特殊なタイプをもつ概念を導入して、その名前部分に別のグラフそのものを埋めこむことを提案しているが、その場合埋めこまれたグラフに対するマッチングの操作の定義に困難が生じる。また、集合を表わす概念についても、同様の問題がありうる。

ＸＣＧは意味表現の枠組であるが、推論の手段としてjoinしが与えられていないために、一般にif-thenで表わされるようなルールによる推論をすることはできない。ＸＣＧのグラフの特徴を活かして、グラフのパターン・マッチングとif-thenルールを組み合わせた推論エンジンのアイディアは[Maruyama86]で述べられている。

グラフのマッチングは一般にはひどく効率の悪いアルゴリズムとなる。joinの効率的なアルゴリズムの開発は今後の課題である。

参考文献

[藤崎79] 藤崎,間下,諸橋,渋谷,鷹尾, "データ・ベース照会システム『ヤチマタ』と名詞句データ模型," 情報処理学会論文誌, Vol.20, No.1, 1979.

[Minsky75] Minsky, M., "A Framework for Representing Knowledge," in: The Psychology of Computer Vision, McGraw-Hill, 1975.

[三吉85] 三吉ほか, "状況意味論に基づく談話理解システムDUALSー租のインプリメンテーションー," 自然言語研究会資料50-7, 1985.

[三吉86] 三吉ほか, "日本語の句構造文法ーJPSG," コンピュータ・ソフトウェア, Vol3., No.4, 1986.

[Montague74] Montague, R., "Proper Treatment of Quantification in Ordinary English," in: Thompson(ed.), Formal Philosophy, Yale University, 1974.

[Pereira83] Pereira, F. "Logic for Natural Language Analysis," SRI techinical Note 275, 1983.

[Schank81] Schank, R. C. and Riesbeck, C. K., Inside Computer Understanding, Lawrence Erlbaum Associates, 1981.

[Sowa84] Sowa, J. F., Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, 1984.

[Sowa85] Sowa, J. F., "Using a Lexicon of Canonical Graphs in a Conceptual Parser," Workshop on the Lexicon, Parsing, and Semantic Interpretations, CUNY Graduate Center, New york, 1985.

[Sowa86] Sowa, J. F. and Way, E. C., "Implementing a Semantic Interpreter Using Conceptual Graphs," IBM Journal of Research and Development, Vol.30, No.1, 1986.

[田中86] 田中ほか, "「論理と自然言語」特集号を編集するにあたって," コンピュータ・ソフトウェア, Vol.3, No.4, 1986.

[丸山85] 丸山, "Conceptual Graphによる談話理解に向けて," 知識工学と人工知能研究会資料41-11, 1985.

[Maruyama86] Maruyama, H., "A Discourse Analysis by Maximal Graph Matching," TRL Research Report TR87-0017, 1986.

[丸山86a] 丸山, "最大グラフ・マッチングによる談話理解," 情報処理学会第33回全国大会予稿集, 1986.

[丸山86b] 丸山, "最大グラフ・マッチングによる文脈解析," ソフトウェア科学会第３回全国大会予稿集, 1986.